

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 [97]:

```
1 # Some resources on Python robot simulation can be found here: https://jyro.readthedocs.io/en/latest/
```

In [98]:

```
1 from ibm_quantum_widgets import CircuitComposer
2 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, Aer, execute
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 [99]:

```

1  # from: https://stackoverflow.com/questions/39298928/play-multiple-sounds-at-the
2
3  # we can use this to play multiple notes (the chord!) at the same time
4
5  from pydub import AudioSegment
6  from pydub.playback import play
7
8  audio0 = AudioSegment.from_file("notes/example.wav") # play as an example
9  # play(audio0)
10
11 audio1 = AudioSegment.from_file("notes/tC.mp3") # note 1 # C played with trumpet
12 audio2 = AudioSegment.from_file("notes/fE.mp3") # note 2 # E played with flute
13 audio3 = AudioSegment.from_file("notes/cG.mp3") # note 3 # G played with cello
14
15 mixed = audio1.overlay(audio2) # combine , superimpose audio files
16 mixed1 = mixed.overlay(audio3) # further combine , superimpose audio 1
17
18 mixed1.export("notes/mixed.mp3", format='mp3') # export mixed audio file
19 play(mixed1) # play mixed audio file
20 # change this line at each time point, so in the end we can get a little piece :
21

```

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpjboysmq5.wav':

```

Duration: 00:00:07.31, 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

```

In [100]:

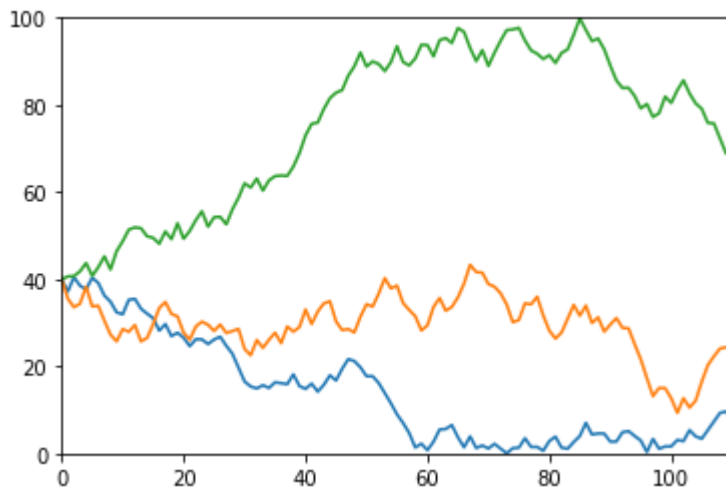
```
1 # bounded random walk:
2 # example from https://stackoverflow.com/questions/46954510/random-walk-series-l
3
4
5 def bounded_random_walk(length, lower_bound, upper_bound, start, end, std):
6     assert (lower_bound <= start and lower_bound <= end)
7     assert (start <= upper_bound and end <= upper_bound)
8
9     bounds = upper_bound - lower_bound
10
11     rand = (std * (np.random.random(length) - 0.5)).cumsum()
12     rand_trend = np.linspace(rand[0], rand[-1], length)
13     rand_deltas = (rand - rand_trend)
14     rand_deltas /= np.max([1, (rand_deltas.max()-rand_deltas.min())/bounds])
15
16     trend_line = np.linspace(start, end, length)
17     upper_bound_delta = upper_bound - trend_line
18     lower_bound_delta = lower_bound - trend_line
19
20     upper_slips_mask = (rand_deltas-upper_bound_delta) >= 0
21     upper_deltas = rand_deltas - upper_bound_delta
22     rand_deltas[upper_slips_mask] = (upper_bound_delta - upper_deltas)[upper_slip
23
24     lower_slips_mask = (lower_bound_delta-rand_deltas) >= 0
25     lower_deltas = lower_bound_delta - rand_deltas
26     rand_deltas[lower_slips_mask] = (lower_bound_delta + lower_deltas)[lower_slip
27
28     return trend_line + rand_deltas
29
30 randomData1 = bounded_random_walk(1000, lower_bound=0, upper_bound =100, start=4
31 randomData2 = bounded_random_walk(1000, lower_bound=0, upper_bound =100, start=4
32 randomData3 = bounded_random_walk(1000, lower_bound=0, upper_bound =100, start=4
```

In [101]:

```

1 plt.figure()
2 # plt.plot(range(randomData.shape[0]), randomData1,randomData2,randomData3)
3 plt.plot(randomData1)
4 plt.plot(randomData2)
5 plt.plot(randomData3)
6 plt.axis([0, 110, 0, 100])
7 plt.show()
8 plt.close()

```



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 [102]:

```

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 qc3 = QuantumCircuit(q, m2, m3, m4); # to reach the target
7 qc4 = QuantumCircuit(q, m2, m3, m4); # to get back to the nest

```

In [103]:

```

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

```

In [104]:

```

1 # let us change the reward position
2 #T = Target("T",0.9,0.5)
3
4 # changing the Target position
5 #T = Target("T",0.2,0.9)
6 #T = Target("T", 0.9, 0.9)
7 T = Target("T", 0.9, 0.5)
8 # for getting back to the beginning
9 T2 = Target("T2",0.2,0.5)

```

In [105]:

```

1 def reward(T, betax, betay):
2     r = round(1 - ((T.x - betax)**2 + (T.y - betay)**2)**0.5, 2)
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 [106]:

```

1 class Robot1:
2     def __init__(self, name, alphax, betax, alphay, betay, gamma, delta):
3         self.name = name
4         self.alphax = alphax
5         self.betax = betax
6         self.alphay = alphay
7         self.betay = betay
8         delta = reward(T, betax, betay)
9         gamma = round(1 - delta, 2)
10        self.gamma = gamma
11        self.delta = delta

```

In [107]:

```

1 # manual intervention needed here to avoid circularity
2 reward(T, 0.2, 0.5) # value of delta

```

Out[107]:

0.3

In [108]:

```

1 # manual intervention needed here to avoid circularity
2 round(1 - reward(T, 0.2, 0.5), 2) # value of gamma

```

Out[108]:

0.7

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

In [109]:

```

1 # (name, alphax, betax, alphay, betay, gamma, delta)
2 R1 = Robot1("R1", 0.8, 0.2, 0.5, 0.5, 0.7, 0.3)

```

In [110]:

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

Out[110]:

(0.7, 0.3)

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 [111]:

```
1 class Robot2:
2     def __init__(self,name,alphax, betax, alphay, betay, gamma, delta):
3         self.name = name
4         self.alphax = alphax
5         self.betax = betax
6         self.alphay = alphay
7         self.betay = betay
8         delta = reward(T, betax, betay)
9         gamma = round(1 - delta, 2)
10        self.gamma = gamma
11        self.delta = delta
```

In [112]:

```
1 reward(T, 0.24, 0.5) # manual intervention needed here to avoid circularity
```

Out[112]:

0.34

In [113]:

```
1 round(1 - reward(T, 0.24, 0.5), 2) # manual intervention needed here to avoid c
```

Out[113]:

0.66

In [114]:

```
1 R2 = Robot2("R2",0.76, 0.24, 0.5, 0.5, 0.66, 0.34) # update by hand this line
```

In [115]:

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

Out[115]:

(0.34, 0.66, 0.76, 0.24, 0.5, 0.5)

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 [116]:

```
1 class Robot3:
2     def __init__(self, name, alphax, betax, alphay, betay, gamma, delta):
3         self.name = name
4         self.alphax = alphax
5         self.betax = betax
6         self.alphay = alphay
7         self.betay = betay
8         delta = reward(T, betax, betay)
9         gamma = round(1 - delta, 2)
10        self.gamma = gamma
11        self.delta = delta
```

In [117]:

```
1 reward(T, 0.19, 0.54) # manual intervention needed here to avoid circularity
```

Out[117]:

0.29

In [118]:

```
1 round(1 - reward(T, 0.19, 0.54), 2) # manual intervention needed here to avoid c
```

Out[118]:

0.71

In [119]:

```
1 R3 = Robot3("R3", 0.8, 0.19, 0.46, 0.54, 0.71, 0.29) # to be updated by hand
```

In [120]:

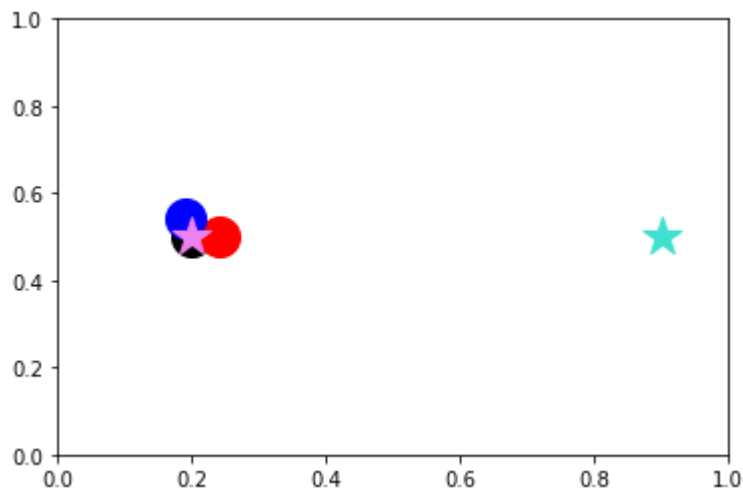
```
1 R3.gamma, R3.delta
```

Out[120]:

(0.71, 0.29)

In [121]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [122]:

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

Out[122]:

```
(0.3, 0.34, 0.29)
```



In [123]:

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

Out[123]:

(0.46, 0.54)

In [124]:

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

In [125]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52        audio2 = AudioSegment.from_file("notes_/fC.mp3")
53        print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56        audio2 = AudioSegment.from_file("notes_/fB.mp3")
57        print("fB")
58     if (R2.betay >= 0.5):
59        audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```
60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):
```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")

```

tD

fD

cD

In [126]:

```

1 mixed_time1_ = audio1.overlay(audio2)           # combine , superimpose audio fil
2 mixed_time1  = mixed_time1_.overlay(audio3)      # further combine , superin
3
4 mixed_time1.export("notes_/mixed_time1.mp3", format='mp3') # export mixed audio
5 play(mixed_time1)                                # play mixed audio file
6 # change this line at each time point, so in the end we can get a little piece :
7

```

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmp9il2k0m3.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

7.25 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 [127]:

```

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 [128]:

```
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 c
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)
```

In [129]:

```
1 R1.alphax, R1.betax, R1.alphay, R1.betay
```

Out[129]:

(0.8, 0.2, 0.5, 0.5)

In [130]:

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

0.3

0.34

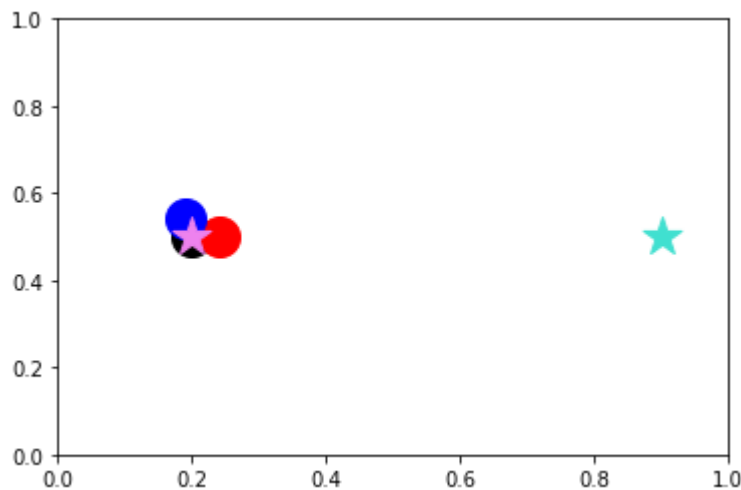
0.29

In [131]:

```

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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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

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 [132]:

```

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(T, R1.betax, R1.betay)
29 R1.gamma = 1 - R1.delta
30 R2.delta = reward(T, R2.betax, R2.betay)
31 R2.gamma = 1 - R2.delta
32 R3.delta = reward(T, R3.betax, R3.betay)
33 R3.gamma = 1 - R3.delta
34 print(R1.delta, R2.delta, R3.delta)

```

SOS

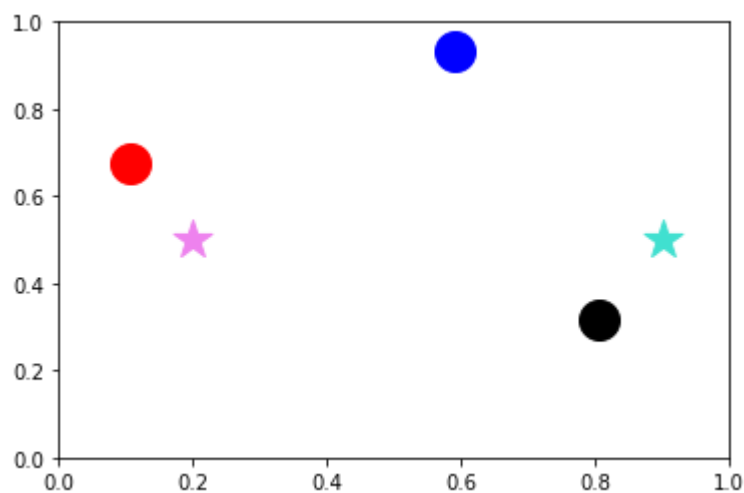
```

the new x-positions for R1 are:  0.193 0.807
the new y-positions for R1 are:  0.683 0.317
the new x-positions for R2 are:  0.894 0.807
the new y-positions for R2 are:  0.325 0.317
the new x-positions for R3 are:  0.409 0.807
the new y-positions for R3 are:  0.068 0.317
0.79 0.19 0.47

```

In [133]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [134]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52         audio2 = AudioSegment.from_file("notes_/fC.mp3")
53         print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56         audio2 = AudioSegment.from_file("notes_/fB.mp3")
57         print("fB")
58     if (R2.betay >= 0.5):
59         audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```
60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):
```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time2_ = audio1.overlay(audio2)           # combine , superimpose audio fi
143     mixed_time2_ = mixed_time2_.overlay(audio3)     # further combine , superi
144
145     mixed_time2.export("notes_/mixed_time2.mp3", format='mp3') # export mixed audi
146     play(mixed_time2)                                     # play mixed audio file
147     # change this line at each time point, so in the end we can get a little piece
148

```

tF

fC#

cG#

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpx7\_bx8bp.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

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

In [135]:

```
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 [136]:

```

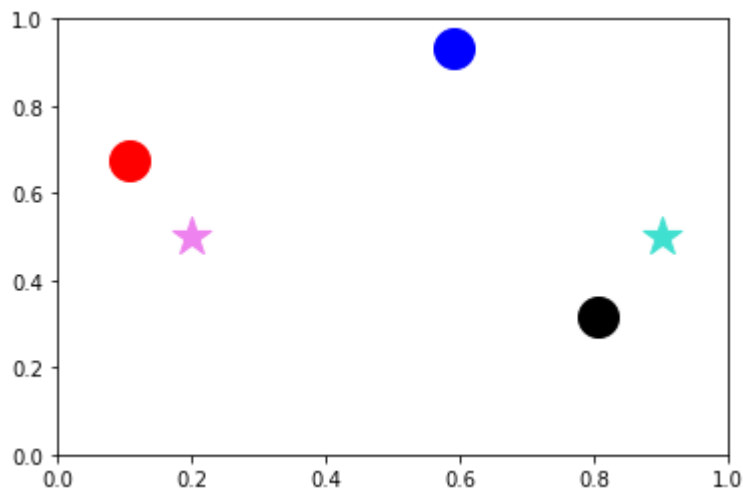
1 if ((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2     print('yuk')
3     if (R1.delta > R2.delta and R1.delta > R3.delta):
4         print('quokka')
5         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later
6         R2.alphax = round(1 - R2.betax, 3)
7 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

```

0.79  
0.19  
0.19

In [137]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [138]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52        audio2 = AudioSegment.from_file("notes_/fC.mp3")
53        print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56        audio2 = AudioSegment.from_file("notes_/fB.mp3")
57        print("fB")
58     if (R2.betay >= 0.5):
59        audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```
60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):
```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time3_ = audio1.overlay(audio2)           # combine , superimpose audio fi
143     mixed_time3 = mixed_time3_.overlay(audio3)      # further combine , superi
144
145     mixed_time3.export("notes_/mixed_time3.mp3", format='mp3') # export mixed audi
146     play(mixed_time3)
147

```

tF

fC#

cG#

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpc7u\_qb5p.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

7.24 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



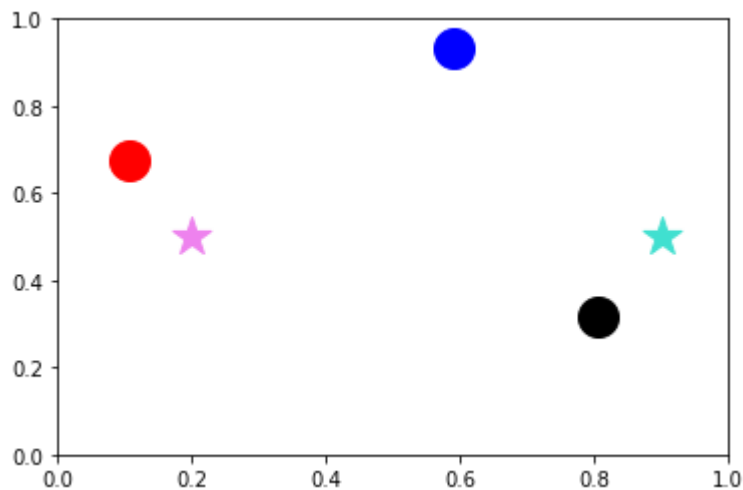
In [139]:

```
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, R1.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, R1.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, R1.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, R1.betay)
29
30 R1.delta = reward(T, R1.betax, R1.betay)
31 R1.gamma = 1 - R1.delta
32 R2.delta = reward(T, R2.betax, R2.betay)
33 R2.gamma = 1 - R2.delta
34 R3.delta = reward(T, R3.betax, R3.betay)
35 R3.gamma = 1 - R3.delta
36 print(R1.delta, R2.delta, R3.delta)
```

0.79 0.19 0.47

In [140]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [141]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52        audio2 = AudioSegment.from_file("notes_/fC.mp3")
53        print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56        audio2 = AudioSegment.from_file("notes_/fB.mp3")
57        print("fB")
58     if (R2.betay >= 0.5):
59        audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```
60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):
```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time4_ = audio1.overlay(audio2) # combine , superimpose audio fi
143     mixed_time4 = mixed_time4_.overlay(audio3) # further combine , superi
144
145     mixed_time4.export("notes_/mixed_time4.mp3", format='mp3') # export mixed audi
146     play(mixed_time4)

```

tF

fC#

cG#

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpwgmxi1j.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  
 7.20 M-A: 0.000 fd= 0 aq= 0KB vq= 0KB sq= 0B f=0/0

7.28 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 [142]:

```

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          qc3.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          qc3.h(q[0])
6      if (R1.alphax >= 0.3) and (R2.alphax < 0.5):
7          print('jungle!')
8          qc3.ry(1.9106332, q[0])
9      if (R1.alphax >= 0.6) and (R2.alphax < 0.7):
10         print('ocean!')
11         qc3.ry(1.2309594, q[0])
12     if (R1.alphay <= 0.2): # else: the qubit sticks with the default value '0'
13         qc3.x(q[1])
14     if (R1.alphay == 0.5): # I have to customize state vectors according to pre
15         qc3.h(q[1])
16     if (R1.alphay >= 0.3) and (R2.alphay < 0.5):
17         print('jungle!')
18         qc3.ry(1.9106332, q[1])
19     if (R1.alphay >= 0.6) and (R2.alphay < 0.7):
20         print('ocean!')
21         qc3.ry(1.2309594, q[1])
22     if (R1.delta == 0.5):
23         qc3.h(q[2])
24     if (R1.delta == 0.6):
25         qc3.h(q[2])
26     if (R1.delta >= 0.7):
27         qc3.x(q[2])
28     if (R1.gamma >= 0.3) and (R2.gamma < 0.5):
29         print('jungle!')
30         qc3.ry(1.9106332, q[2])
31     if (R1.gamma >= 0.6) and (R2.gamma < 0.7):
32         print('ocean!')
33         qc3.ry(1.2309594, q[2])
34 # elif (R1.delta > R2.delta) and (R1.delta > R3.delta): # February 13: NO!!!
35 elif (R2.delta > R1.delta) and (R2.delta > R3.delta):
36     print('dog')
37     if (R2.alphax < 0.3): # I have to customize state vectors according to prec
38         qc3.x(q[0])      # just using the NOT gate as a test
39     if (R2.alphax == 0.5): # I have to customize state vectors according to pre
40         qc3.h(q[0])
41     if (R2.alphax >= 0.3) and (R1.alphax < 0.5):
42         print('jungle!')
43         qc3.ry(1.9106332, q[0])
44     if (R2.alphax >= 0.6) and (R1.alphax < 0.7):
45         print('ocean!')
46         qc3.ry(1.2309594, q[0])
47     if (R2.alphay <= 0.2): # else: the qubit sticks with the default value '0'
48         qc3.x(q[1])
49     if (R2.alphay == 0.5): # I have to customize state vectors according to pre
50         qc3.h(q[1])
51     if (R2.alphay >= 0.3) and (R1.alphay < 0.5):
52         print('jungle!')
53         qc3.ry(1.9106332, q[1])
54     if (R2.alphay >= 0.6) and (R1.alphay < 0.7):
55         print('ocean!')
56         qc3.ry(1.2309594, q[1])
57     if (R2.delta == 0.5):
58         qc3.h(q[2])
59     if (R2.delta == 0.6):

```

```

60     qc3.h(q[2])
61     if (R2.delta >= 0.7):
62         qc3.x(q[2])
63     if (R2.gamma >= 0.3) and (R1.gamma < 0.5):
64         print('jungle!')
65         qc3.ry(1.9106332, q[2])
66     if (R2.gamma >= 0.6) and (R1.gamma < 0.7):
67         print('ocean!')
68         qc3.ry(1.2309594, q[2])
69 else:
70     print('cat') # I made some tests to check the IF conditions
71     if (R3.alphax < 0.3):
72         qc3.x(q[0])
73     if (R3.alphax == 0.5):
74         qc3.h(q[0])
75     if (R3.alphax >= 0.3) and (R3.alphax < 0.5):
76         print('jungle!')
77         qc3.ry(1.9106332, q[0])
78     if (R3.alphax >= 0.6) and (R3.alphax < 0.7):
79         print('ocean!')
80         qc3.ry(1.2309594, q[0])
81     if (R3.alphay < 0.3):
82         qc3.x(q[1])
83     if (R3.alphay == 0.5):
84         qc3.h(q[1])
85     if (R3.alphay >= 0.3) and (R3.alphay < 0.5):
86         print('jungle!')
87         qc3.ry(1.9106332, q[1])
88     if (R3.alphay >= 0.6) and (R3.alphay < 0.7):
89         print('ocean!')
90         qc3.ry(1.2309594, q[1])
91     if (R3.delta == 0.5):
92         qc3.h(q[2])
93     if (R3.delta == 0.6):
94         qc3.h(q[2])
95     if (R3.delta >= 0.7):
96         qc3.x(q[2])
97     if (R3.gamma >= 0.3) and (R3.gamma < 0.5):
98         print('jungle!')
99         qc3.ry(1.9106332, q[2])
100    if (R3.gamma >= 0.6) and (R3.gamma < 0.7):
101        print('ocean!')
102        qc3.ry(1.2309594, q[2])

```

jungle!  
ocean!

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.

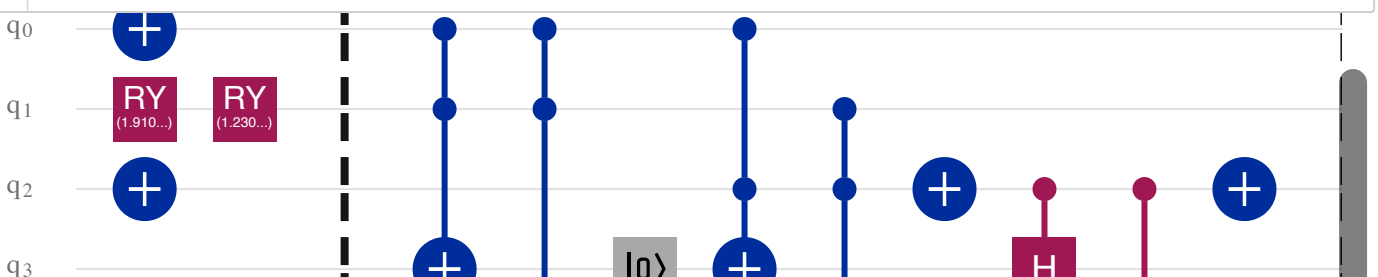


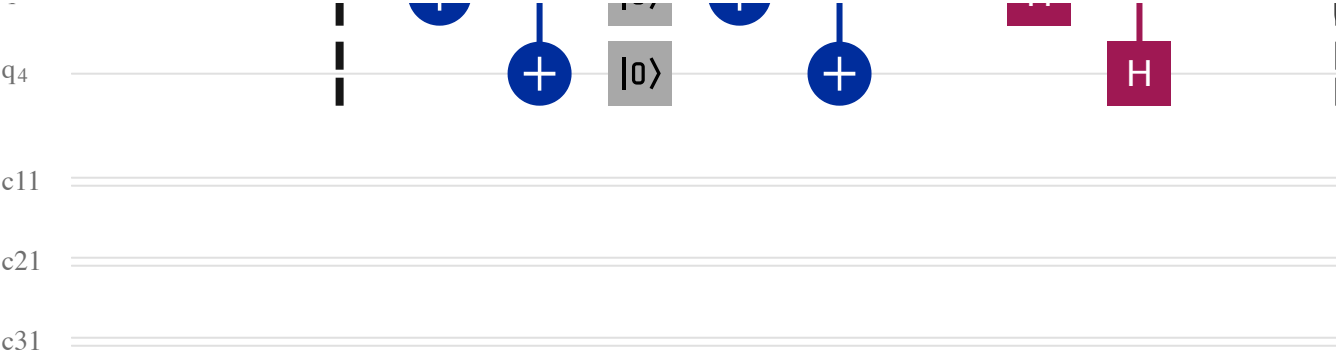
In [143]:

```

1  # this is the core code, and it is unchanged across time
2
3  qc3.barrier(q)
4  qc3.ccx(q[0],q[1],q[3])
5  qc3.ccx(q[0],q[1],q[4])
6
7  qc3.reset(q[3]);
8  qc3.reset(q[4]);
9
10 qc3.ccx(q[0],q[2],q[3])
11 qc3.ccx(q[1],q[2],q[4])
12
13 qc3.x(q[2])
14
15 qc3.ch(q[2],q[3])
16 qc3.ch(q[2],q[4])
17
18 qc3.x(q[2])
19
20 qc3.barrier(q)
21
22 # perform measurements and store them in classical bits
23
24 qc3.measure(q[2],m2[0])
25 qc3.measure(q[3],m3[0])
26 qc3.measure(q[4],m4[0])
27
28 # visualization of the circuit
29
30 draw_circuit(qc3)
31
32 # definition of quantum simulator
33
34 simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_simu
35 qc3 = transpile(qc3, simulator)
36 cc = collections.Counter()
37
38 # Run and get counts
39 result = simulator.run(qc3, shots=1024).result()
40 counts = result.get_counts(qc3)
41 counts2 = counts.most_frequent() # does not work if multiple states have the same
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(qc3, shots=10, memory=True).result()
48 memory = result.get_memory(qc3)
49 print(memory)
50 plot_histogram(counts, title='outcomes')
51 # TAKE the TWO more present outcomes

```





In [54]:

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

In [144]:

```

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 = 1024 # AT HAND ONLY FOR NOW
7
8 arr2 = ['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

```

```

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

```

```

-----
IndexError                                Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_3140/4264151028.py in <
module>
    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 [145]:

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

Out[145]:

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

In [146]:

```
1 # array 2
2 arr2
```

Out[146]:

```
['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).

Position for  $R_1$ :

In [147]:

```

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     # change of January 38: I'm substituting [0] with [1] and vice versa, becau
12     if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
13         print("bla")
14         R1.alphax = 0.3
15         R1.betax = 0.7
16     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
17         print("gulp")
18         R1.alphax = 0.7
19         R1.betax = 0.3
20     elif (arr1[1] == arr2[0]) and (arr1[1] == '1') and (weight1 == weight2 or n
21         print("stra-gulp")
22         R1.alphax = 0.5 # change temporarily made on January 24: random generat
23         R1.betax = 0.5 # same as above
24     elif (arr1[1] == arr2[0]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
25         print("thunderstorm!")
26         R1.alphax = 0
27         R1.betax = 1
28     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
29         print("avalanche!")
30         R1.alphax = 0.1
31         R1.betax = 0.9
32     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
33         print("earthquake!")
34         R1.alphax = 0.9
35         R1.betax = 0.1
36     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
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[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
42         print("bla 2")
43         R1.alphax = 0.7 # the opposite??
44         R1.betax = 0.3 # the opposite??
45     elif (arr1[1] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
46         print("gulp 2")
47         R1.alphax = 0.3 # the opposite??
48         R1.betax = 0.7 # the opposite??
49     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
50         print("stra-gulp 2")
51         R1.alphax = 0.5 # change temporarily made on January 24: random generat
52         R1.betax = 0.5 # change temporarily made on January 24: random generato
53     # different outcomes
54     elif arr1[1] != arr2[1]: # January 23
55         print("blue")
56         if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
57             print("google 1")
58             R1.alphax = 0.5 # change temporarily made on January 24: random gen
59             R1.betax = 0.5 # change temporarily made on January 24: random gene

```

```

60     if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
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[1] == '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
75             print("news")
76             R1.alphax = 0.2 #
77             R1.betax = 0.8 #
78         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
79             print("idea")
80             R1.alphax = 0.8 #
81             R1.betax = 0.2 #
82         if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
83             print("news")
84             R1.alphax = 0.1 #
85             R1.betax = 0.9 #
86         if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
87             print("idea")
88             R1.alphax = 0.9 #
89             R1.betax = 0.1 #
90     if (arr1[1] == '0') and (arr2[1] == '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
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
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
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
112             print("hand2")
113             R1.alphax = 0.1 # 0.1
114             R1.betax = 0.9 # 0.9
115     # y part
116
117     # change of January 26
118
119     if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
120         print("bla")

```

```

121     R1.alphay = 0.3
122     R1.betay = 0.7
123     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
124         print("gulp")
125         R1.alphay = 0.7
126         R1.betay = 0.3
127     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
128         print("stra-gulp")
129         R1.alphay = 0.5 # change temporarily made on January 24: random generat
130         R1.betay = 0.5 # same as above
131     elif (arr1[0] == arr2[1]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
132         print("thunderstorm!")
133         R1.alphay = 0
134         R1.betay = 1
135     elif (arr1[0] == arr2[1]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
136         print("avalanche!")
137         R1.alphay = 0.1
138         R1.betay = 0.9
139     elif (arr1[0] == arr2[1]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
140         print("earthquake!")
141         R1.alphay = 0.9
142         R1.betay = 0.1
143     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
144         print("avalanche bis!")
145         R1.alphay = 0.1 # the same also in this case
146         R1.betay = 0.9 # the same also in this case
147     # same = outcome 0 # January 23
148     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
149         print("bla 2")
150         R1.alphay = 0.7 # the opposite??
151         R1.betay = 0.3 # the opposite??
152     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
153         print("gulp 2")
154         R1.alphay = 0.3 # the opposite??
155         R1.betay = 0.7 # the opposite??
156     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
157         print("stra-gulp 2")
158         R1.alphay = 0.5 # change temporarily made on January 24: random generat
159         R1.betay = 0.5 # change temporarily made on January 24: random generato
160     # different outcomes
161     elif arr1[0] != arr2[0]: # January 23
162         print("blue")
163         if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
164             print("google 1")
165             R1.alphay = 0.5 # change temporarily made on January 24: random gen
166             R1.betay = 0.5 # change temporarily made on January 24: random gene
167         if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
168             # include the case of a very small difference!
169             print("uffdah")
170             R1.alphay = 0.5
171             R1.betay = 0.5
172         if (arr1[0] == '1' and arr2[0] == '0'):
173             if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
174                 print("abc")
175                 R1.alphay = 0.3
176                 R1.betay = 0.7
177             if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
178                 print("bca")
179                 R1.alphay = 0.7
180                 R1.betay = 0.3
181             if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no

```

```

182         print("news")
183         R1.alphay = 0.2 #
184         R1.betay = 0.8 #
185     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
186         print("idea")
187         R1.alphay = 0.8 #
188         R1.betay = 0.2 #
189     if (arr1[0] == '0') and (arr2[1] == '1'):
190         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
191             print("bac")
192             R1.alphay = 0.7
193             R1.betay = 0.3
194         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
195             print("cba")
196             R1.alphay = 0.3
197             R1.betay = 0.7
198         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
199             print("brain")
200             R1.alphay = 0.7 # 0.9
201             R1.betay = 0.3 # 0.1
202         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
203             print("hand")
204             R1.alphay = 0.3 # 0.1
205             R1.betay = 0.7 # 0.9
206         if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
207             print("brain2")
208             R1.alphay = 0.9 # 0.9
209             R1.betay = 0.1 # 0.1
210         if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
211             print("hand2")
212             R1.alphay = 0.1 # 0.1
213             R1.betay = 0.9 # 0.9

```

Position for  $R_2$



In [148]:

```

1  if (R2.delta > R1.delta) and (R2.delta > R3.delta):
2      # if R2 didn't entered 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 = 0.9
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 = 0.2 # or: 0.3
76         R2.betax = 0.8 # or: 0.7
77     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
78         print("idea")
79         R2.alphax = 0.8 #
80         R2.betax = 0.2 #
81     if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
82         print("news")
83         R2.alphax = 0.1 # or: 0.3
84         R2.betax = 0.9 # or: 0.7
85     if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
86         print("idea")
87         R2.alphax = 0.9 #
88         R2.betax = 0.1 #
89 if (arr1[1] == '0') and (arr2[1] == '1'):
90     print("sigh")
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
100 # y part
101 # change of January 26
102
103 # to be modified: R1 to R2
104
105 if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
106     print("bla")
107     R2.alphay = 0.3
108     R2.betay = 0.7
109 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
110     print("gulp")
111     R2.alphay = 0.7
112     R2.betay = 0.3
113 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
114     print("stra-gulp")
115     R2.alphay = 0.5 # change temporarily made on January 24: random generat
116     R2.betay = 0.5 # same as above
117 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
118     print("thunderstorm!")
119     R2.alphay = 0
120     R2.betay = 1

```

```

121 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
122     print("avalanche!")
123     R2.alphay = 0.1
124     R2.betay = 0.9
125 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
126     print("earthquake!")
127     R2.alphay = 0.9
128     R2.betay = 0.1
129 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
130     print("avalanche bis!")
131     R2.alphay = 0.1 # the same also in this case
132     R2.betay = 0.9 # the same also in this case
133 # same = outcome 0 # January 23
134 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
135     print("bla 2")
136     R2.alphay = 0.7 # the opposite??
137     R2.betay = 0.3 # the opposite??
138 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
139     print("gulp 2")
140     R2.alphay = 0.3 # the opposite??
141     R2.betay = 0.7 # the opposite??
142 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
143     print("stra-gulp 2")
144     R2.alphay = 0.5 # change temporarily made on January 24: random generat
145     R2.betay = 0.5 # change temporarily made on January 24: random generato
146 # different outcomes
147 elif arr1[0] != arr2[1]: # January 23
148     print("blue")
149     if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
150         print("google 1")
151         R2.alphay = 0.5 # change temporarily made on January 24: random gen
152         R2.betay = 0.5 # change temporarily made on January 24: random gene
153     if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
154         # include the case of a very small difference!
155         print("uffdah")
156         R2.alphay = 0.5
157         R2.betay = 0.5
158     if (arr1[0] == '1' and arr2[0] == '0'):
159         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
160             print("abc")
161             R2.alphay = 0.3
162             R2.betay = 0.7
163         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
164             print("bca")
165             R2.alphay = 0.7
166             R2.betay = 0.3
167         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
168             print("news")
169             R2.alphay = 0.2 # January 26
170             R2.betay = 0.8 #
171         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
172             print("idea")
173             R2.alphay = 0.8 #
174             R2.betay = 0.2 #
175     if (arr1[0] == '0') and (arr2[1] == '1'):
176         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
177             print("bac")
178             R2.alphay = 0.7
179             R2.betay = 0.3
180         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
181             print("cba")

```

```
182         R2.alphay = 0.3
183         R2.betay = 0.7
184     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
185         print("brain")
186         R2.alphay = 0.7 # 0.9
187         R2.betay = 0.3 # 0.1
188     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
189         print("hand")
190         R2.alphay = 0.3 # 0.1
191         R2.betay = 0.7 # 0.9
192     if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
193         print("brain")
194         R2.alphay = 0.9 # 0.9
195         R2.betay = 0.1 # 0.1
196     if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
197         print("hand")
198         R2.alphay = 0.1 # 0.1
199         R2.betay = 0.9 # 0.9
```

stra-gulp  
stra-gulp

Position for  $R_3$

In [149]:

```

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

```

```

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

```

```

121 # to be modified: R1 to R3
122
123 if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
124     print("bla")
125     R3.alphay = 0.3
126     R3.betay = 0.7
127 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
128     print("gulp")
129     R3.alphay = 0.7
130     R3.betay = 0.3
131 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
132     print("stra-gulp")
133     R3.alphay = 0.5 # change temporarily made on January 24: random generat
134     R3.betay = 0.5 # same as above
135 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
136     print("thunderstorm!")
137     R3.alphay = 0
138     R3.betay = 1
139 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
140     print("avalanche!")
141     R3.alphay = 0.1
142     R3.betay = 0.9
143 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
144     print("earthquake!")
145     R3.alphay = 0.9
146     R3.betay = 0.1
147 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
148     print("avalanche bis!")
149     R3.alphay = 0.1 # the same also in this case
150     R3.betay = 0.9 # the same also in this case
151 # same = outcome 0 # January 23
152 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
153     print("bla 2")
154     R3.alphay = 0.7 # the opposite??
155     R3.betay = 0.3 # the opposite??
156 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
157     print("gulp 2")
158     R3.alphay = 0.3 # the opposite??
159     R3.betay = 0.7 # the opposite??
160 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
161     print("stra-gulp 2")
162     R3.alphay = 0.5 # change temporarily made on January 24: random generat
163     R3.betay = 0.5 # change temporarily made on January 24: random generato
164 # different outcomes
165 elif arr1[0] != arr2[0]: # January 23
166     print("blue")
167     if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
168         print("google 1")
169         R3.alphay = 0.5 # change temporarily made on January 24: random gen
170         R3.betay = 0.5 # change temporarily made on January 24: random gene
171     if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
172         # include the case of a very small difference!
173         print("uffdah")
174         R3.alphay = 0.5
175         R3.betay = 0.5
176     if (arr1[0] == '1' and arr2[0] == '0'):
177         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
178             print("abc")
179             R3.alphay = 0.3
180             R3.betay = 0.7
181         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):

```



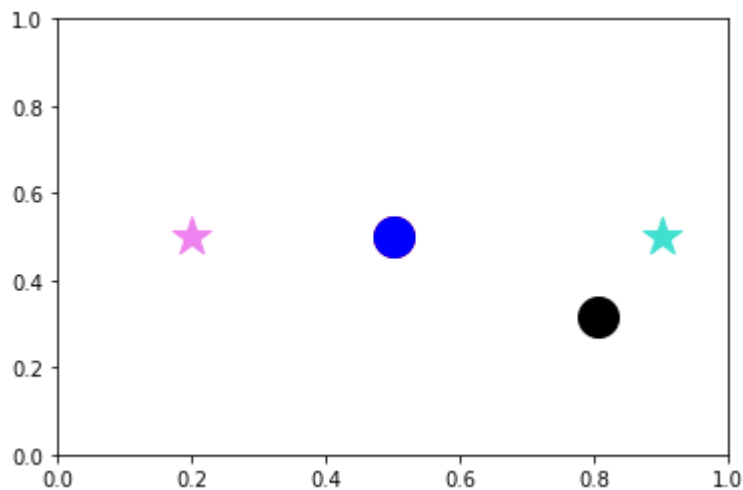
```
182         print("bca")
183         R3.alphay = 0.7
184         R3.betay = 0.3
185     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
186         print("news")
187         R3.alphay = 0.2 #
188         R3.betay = 0.8 #
189     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
190         print("idea")
191         R3.alphay = 0.8 #
192         R3.betay = 0.2 #
193     if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
194         print("news")
195         R3.alphay = 0.1 #
196         R3.betay = 0.9 #
197     if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
198         print("idea")
199         R3.alphay = 0.9 #
200         R3.betay = 0.1 #
201 if (arr1[0] == '0') and (arr2[1] == '1'):
202     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
203         print("bac")
204         R3.alphay = 0.7
205         R3.betay = 0.3
206     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
207         print("cba")
208         R3.alphay = 0.3
209         R3.betay = 0.7
210     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
211         print("brain")
212         R3.alphay = 0.7 # 0.9
213         R3.betay = 0.3 # 0.1
214     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
215         print("hand")
216         R3.alphay = 0.3 # 0.1
217         R3.betay = 0.7 # 0.9
218     if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
219         print("brain")
220         R3.alphay = 0.9 # 0.9
221         R3.betay = 0.1 # 0.1
222     if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
223         print("hand")
224         R3.alphay = 0.1 # 0.1
225         R3.betay = 0.9 # 0.9
```

stra-gulp  
stra-gulp



In [150]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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

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 [151]:

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

Out[151]:

(0.79, 0.19, 0.47)

In [152]:

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

0.79

0.6

0.6

In [153]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52        audio2 = AudioSegment.from_file("notes_/fC.mp3")
53        print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56        audio2 = AudioSegment.from_file("notes_/fB.mp3")
57        print("fB")
58     if (R2.betay >= 0.5):
59        audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```

60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):

```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time5_ = audio1.overlay(audio2)           # combine , superimpose audio fi
143     mixed_time5 = mixed_time5_.overlay(audio3)      # further combine , superi
144
145     mixed_time5.export("notes_/mixed_time5.mp3", format='mp3') # export mixed audi
146     play(mixed_time5)

```

tF

fA

cA

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpfvhgm19e.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

7.22 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 [154]:

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

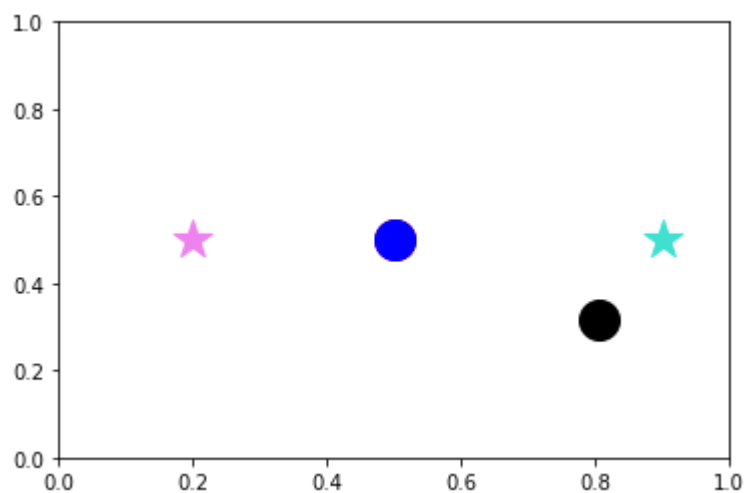
In [155]:

```
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, R1.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, R1.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, R1.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, R1.betay)
29
30 R1.delta = reward(T, R1.betax, R1.betay)
31 R1.gamma = 1 - R1.delta
32 R2.delta = reward(T, R2.betax, R2.betay)
33 R2.gamma = 1 - R2.delta
34 R3.delta = reward(T, R3.betax, R3.betay)
35 R3.gamma = 1 - R3.delta
36 print(R1.delta, R2.delta, R3.delta)
```

0.79 0.6 0.6

In [156]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [157]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52        audio2 = AudioSegment.from_file("notes_/fC.mp3")
53        print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56        audio2 = AudioSegment.from_file("notes_/fB.mp3")
57        print("fB")
58     if (R2.betay >= 0.5):
59        audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```



```
60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):
```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time6_ = audio1.overlay(audio2)           # combine , superimpose audio fi
143     mixed_time6 = mixed_time6_.overlay(audio3)      # further combine , superi
144
145     mixed_time6.export("notes_/mixed_time6.mp3", format='mp3') # export mixed audi
146     play(mixed_time6)

```

tF

fA

cA

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmp3uwuxgqh.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

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

In [158]:

```
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 [159]:

```

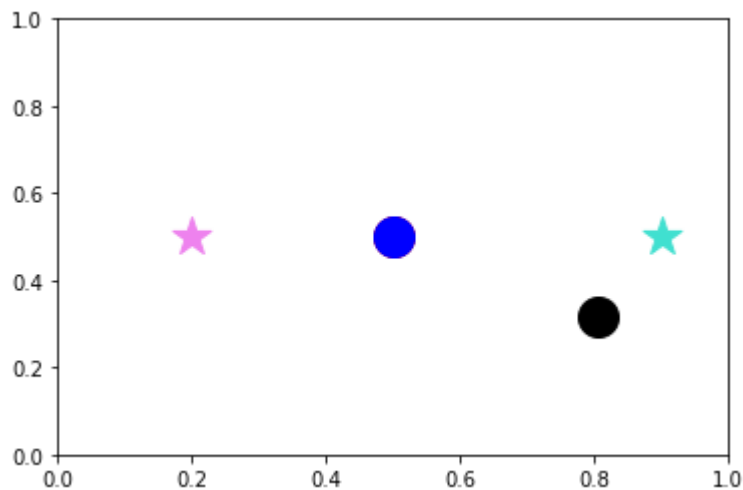
1  if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2      print('yuk')
3      if (R1.delta > R2.delta and R1.delta > R3.delta):
4          print('quokka')
5          R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
6          R2.alphax = round(1 - R2.betax, 3)
7          R2.betay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

```

0.6  
0.6  
0.6

In [160]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [161]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52        audio2 = AudioSegment.from_file("notes_/fC.mp3")
53        print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56        audio2 = AudioSegment.from_file("notes_/fB.mp3")
57        print("fB")
58     if (R2.betay >= 0.5):
59        audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```

60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):

```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time7_ = audio1.overlay(audio2)           # combine , superimpose audio fi
143     mixed_time7 = mixed_time7_.overlay(audio3)      # further combine , superi
144
145     mixed_time7.export("notes_/mixed_time7.mp3", format='mp3') # export mixed audi
146     play(mixed_time7)

```

tF

fA

cA

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmp7vs88g99.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

7.24 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

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

In [162]:

```

1  if (R1.delta > R2.delta) and (R1.delta > R2.delta): # now, brutally come close t
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.alphax + np.random.uniform(0,0.1), 3)
8      R2.betay = round(1 - R2.alphax, 3)
9      # R3
10     R3.alphax = round(R1.alphax + np.random.uniform(0,0.1), 3)
11     R3.betax = round(1 - R2.alphax, 3)
12     R3.alphay = round(R1.alphax + np.random.uniform(0,0.1), 3)
13     R3.betay = round(1 - R2.alphax, 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.alphax + np.random.uniform(0,0.1), 3)
26     R3.betay = round(1 - R2.alphax, 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.alphax + np.random.uniform(0,0.1), 3)
39     R2.betay = round(1 - R2.alphax, 3)

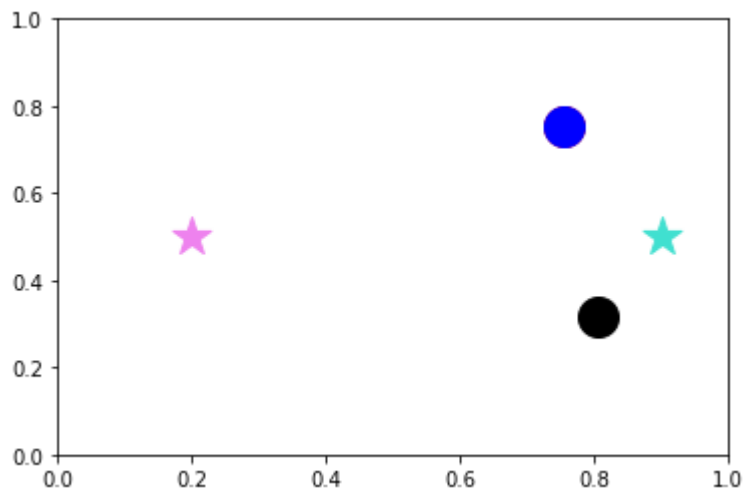
```

ciao



In [163]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [164]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52         audio2 = AudioSegment.from_file("notes_/fC.mp3")
53         print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56         audio2 = AudioSegment.from_file("notes_/fB.mp3")
57         print("fB")
58     if (R2.betay >= 0.5):
59         audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```
60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):
```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time8_ = audio1.overlay(audio2)           # combine , superimpose audio fi
143     mixed_time8 = mixed_time8_.overlay(audio3)      # further combine , superi
144
145     mixed_time8.export("notes_/mixed_time8.mp3", format='mp3') # export mixed audi
146     play(mixed_time8)

```

tF

fG

cG

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpouq8cxg5.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

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

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

New reward amplitude probabilities for  $R_1$ :

In [165]:

```

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

```

0.79

New reward amplitude probabilities for  $R_2$ :

In [166]:

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

0.71

New reward amplitude probabilities for  $R_3$ :

In [167]:

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

0.71

In [168]:

```
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 [169]:

```

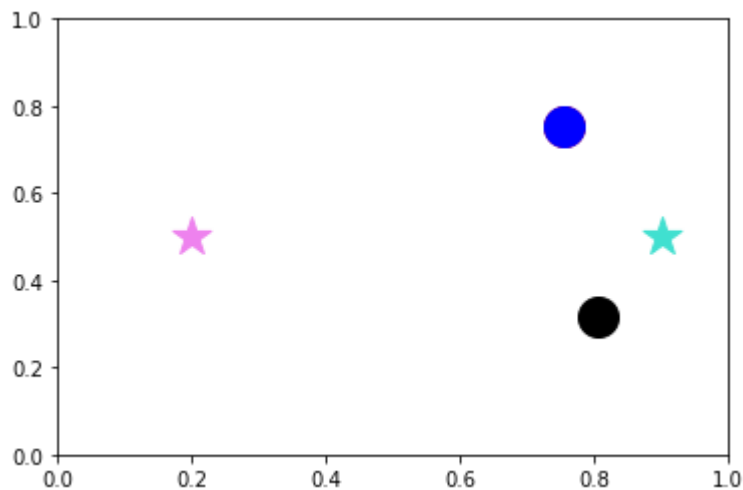
1
2 if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
3     print('yuk')
4     if (R1.delta > R2.delta and R1.delta > R3.delta):
5         print('quokka')
6         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
7         R2.alphax = round(1 - R2.betax, 3)
8         R2.betay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

```

0.71  
0.71  
0.71

In [170]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [171]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52        audio2 = AudioSegment.from_file("notes_/fC.mp3")
53        print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56        audio2 = AudioSegment.from_file("notes_/fB.mp3")
57        print("fB")
58     if (R2.betay >= 0.5):
59        audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```



```
60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):
```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time9_ = audio1.overlay(audio2)           # combine , superimpose audio fi
143     mixed_time9_ = mixed_time9_.overlay(audio3)     # further combine , superi
144
145     mixed_time9.export("notes_/mixed_time9.mp3", format='mp3') # export mixed audi
146     play(mixed_time9)

```

tF

fG

cG

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpsam799cd.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

7.29 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 [172]:

```

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  $\frac{1}{\sqrt{2}}(|1111\rangle + |0000\rangle)$ , 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  $\leq 0.1$ ).

In [173]:

```
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);
```

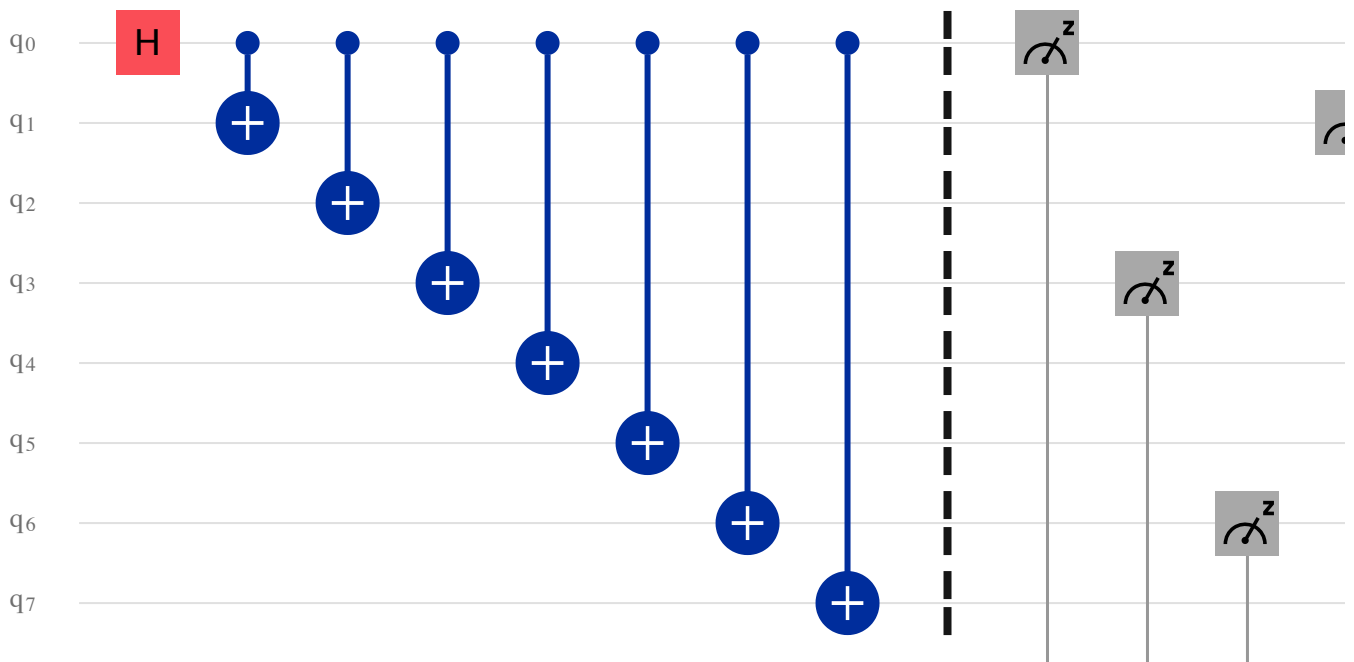
In [174]:

```

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')

```

cometa





```
{'1 1 1 1 1 1': 517, '0 0 0 0 0 0': 507}
1 1 1 1 1 1
```

In [175]:

```
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
```

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

Define the 'new 0':

In [176]:

```
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
```

gosh

Define the 'new 1':

In [177]:

```
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
```

gosh

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

In [178]:

```

1  if (arr_GHZ[0] == '0'):
2      R1.alphax = new_zero_alphax
3      R1.betax = new_zero_betax
4  if (arr_GHZ[1] == '0'):
5      R1.alphay = new_zero_alphay
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_alphay
12     R2.betay = new_zero_betay
13 if (arr_GHZ[3] == '0'):
14     R2.alphax = new_zero_alphax
15     R2.betax = new_zero_betax
16 if (arr_GHZ[4] == '0'):
17     R3.alphax = new_zero_alphax
18     R3.betax = new_zero_betax
19 if (arr_GHZ[5] == '0'):
20     R3.alphay = new_zero_alphay
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_alphay
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_alphay
35     R2.betay = new_one_betay
36 if (arr_GHZ[3] == '1'):
37     R2.alphax = new_one_alphax
38     R2.betax = new_one_betax
39 if (arr_GHZ[4] == '1'):
40     R3.alphax = new_one_alphax
41     R3.betax = new_one_betax
42 if (arr_GHZ[5] == '1'):
43     R3.alphay = new_one_alphay
44     R3.betay = new_one_betay
45

```

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 [179]:

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

0.7

New reward for  $R_2$ :

In [180]:

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

0.7

New reward for  $R_3$ :

In [181]:

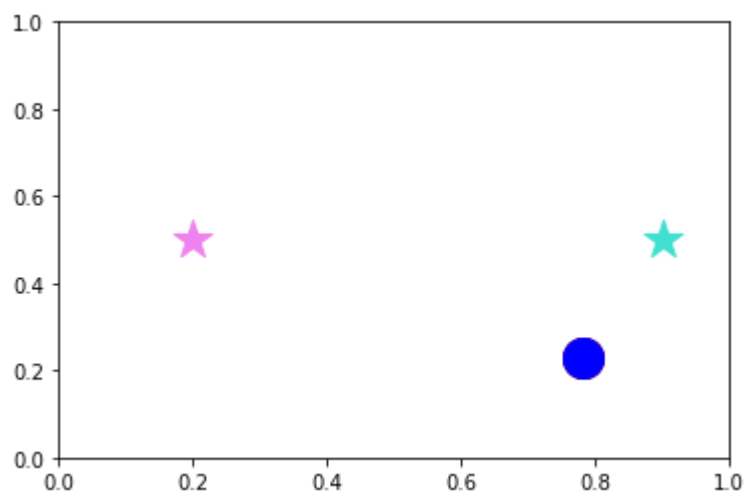
```
1 R3.delta = reward(T,R3.betax,R3.betay)
2 R3.gamma = round((1 - R3.delta),2)
3 print(R3.delta)
```

0.7



In [182]:

```
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
23 plt.axis([0, 1, 0, 1])
24 plt.show()
25 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 [183]:

```

1
2 # audio 1, R_1
3
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
48 # audio 2, R_2
49
50 if (R2.betax == 0):
51     if (R2.betay == 0.5):
52         audio2 = AudioSegment.from_file("notes_/fC.mp3")
53         print("fC")
54 if (R2.betax > 0 and R2.betax <= 0.17):
55     if (R2.betay < 0.5):
56         audio2 = AudioSegment.from_file("notes_/fB.mp3")
57         print("fB")
58     if (R2.betay >= 0.5):
59         audio2 = AudioSegment.from_file("notes_/fC#.mp3")

```

```

60     print("fC#")
61 if (R2.betax > 0.17 and R2.betax <= 0.3):
62     if (R2.betay < 0.5):
63         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
64         print("fA#")
65     if (R2.betay >= 0.5):
66         audio2 = AudioSegment.from_file("notes_/fD.mp3")
67         print("fD")
68 if (R2.betax > 0.3 and R2.betax <= 0.5):
69     if (R2.betay < 0.5): # (R1.betay == 1):
70         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
71         print("fD#")
72     if (R2.betay >= 0.5):
73         audio2 = AudioSegment.from_file("notes_/fA.mp3")
74         print("fA")
75 if (R2.betax > 0.5 and R2.betax <= 0.64):
76     if (R2.betay < 0.5):
77         audio2 = AudioSegment.from_file("notes_/fE.mp3")
78         print("fE")
79     if (R2.betay >= 0.5):
80         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
81         print("fG#")
82 if (R2.betax > 0.64 and R2.betax <= 0.84):
83     if (R2.betay < 0.5):
84         audio2 = AudioSegment.from_file("notes_/fF.mp3")
85         print("fF")
86     if (R2.betay >= 0.5):
87         audio2 = AudioSegment.from_file("notes_/fG.mp3")
88         print("fG")
89 if (R2.betax > 0.84 and R2.betax <= 1):
90     #if (R2.betay == 0.5):
91     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
92     print("fF#")
93
94
95 # audio 3, R_3
96
97
98 if (R3.betax == 0):
99     if (R3.betay == 0.5):
100         audio3 = AudioSegment.from_file("notes_/cC.mp3")
101         print("cC")
102 if (R3.betax > 0 and R3.betax <= 0.17):
103     if (R3.betay < 0.5):
104         audio3 = AudioSegment.from_file("notes_/cB.mp3")
105         print("cB")
106     if (R3.betay >= 0.5):
107         audio3 = AudioSegment.from_file("notes_/cC#.mp3")
108         print("cC#")
109 if (R3.betax > 0.17 and R3.betax <= 0.3):
110     if (R3.betay < 0.5):
111         audio3 = AudioSegment.from_file("notes_/cA#.mp3")
112         print("cA#")
113     if (R3.betay >= 0.5):
114         audio3 = AudioSegment.from_file("notes_/cD.mp3")
115         print("cD")
116 if (R3.betax > 0.3 and R3.betax <= 0.5):
117     if (R3.betay < 0.5):
118         audio3 = AudioSegment.from_file("notes_/cD#.mp3")
119         print("cD#")
120     if (R3.betay >= 0.5):

```

```

121     audio3 = AudioSegment.from_file("notes_/cA.mp3")
122     print("cA")
123     if (R3.betax > 0.5 and R3.betax <= 0.64):
124         if (R3.betay < 0.5):
125             audio3 = AudioSegment.from_file("notes_/cE.mp3")
126             print("cE")
127         if (R3.betay >= 0.5):
128             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
129             print("cG#")
130     if (R3.betax > 0.64 and R3.betax <= 0.84):
131         if (R3.betay < 0.5):
132             audio3 = AudioSegment.from_file("notes_/cF.mp3")
133             print("cF")
134         if (R3.betay >= 0.5):
135             audio3 = AudioSegment.from_file("notes_/cG.mp3")
136             print("cG")
137     if (R3.betax > 0.84 and R3.betax <= 1):
138         #if (R3.betay == 0.5):
139             audio3 = AudioSegment.from_file("notes_/cF#.mp3")
140             print("cF#")
141
142     mixed_time10_ = audio1.overlay(audio2)           # combine , superimpose audio f
143     mixed_time10_ = mixed_time10_.overlay(audio3)    # further combine , supe
144
145     mixed_time10.export("notes_/mixed_time10.mp3", format='mp3') # export mixed au
146     play(mixed_time10)

```

tF

fF

cF

Could not import the PyAudio C module '\_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p\_w0000gn/T/tmpa3zbxihy.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

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

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

In [184]:

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

Out[184]:

(0.7, 0.7, 0.7)

In [88]:

```
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 [185]:

```

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

```

0.7  
0.7  
0.7