Student Name: Anthony and Shubh

Assignment: RECF VR Virtual Skills Online Challenge

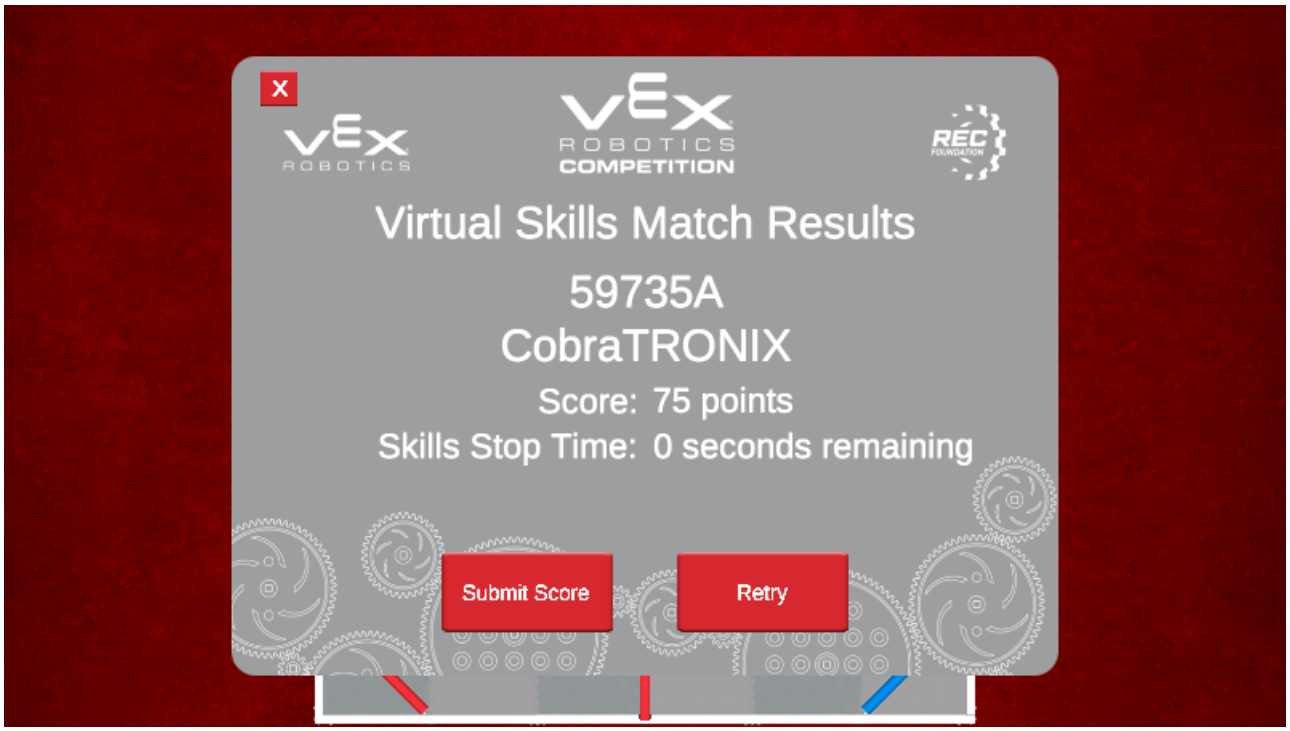
Notes: This code by 59735A Cobratronix is a program based on the OODA loop, a decision-making process created by fighter pilot John Boyd. It stands for Observe, Orient, Decide, and Act. It does this using the GPS Sensor to get its current location. It then takes that information and uses trigonometry and the Pythagorean theorem to create a vector to a location. It then loops through the middle triballs to find the closest triball and target point to its current location.

Playground: VRC Virtual Skills - Over Under

Project Name: VirtualSkillsChallenge

Project Type: Python

Date: Wed Jan 31 2024



```
1      #region VEXcode Generated Robot Configuration
2      import math
3      import random
4      from vexcode_vrc import *
5      from vexcode_vrc.events import get_Task_func
6
7      # Brain should be defined by default
8      brain=Brain()
9
10     drivetrain = Drivetrain("drivetrain", 0)
11     arm_motor = Motor("ArmMotor", 3)
12     rotation = Rotation("Rotation", 7)
13     intake_motor = Motor("IntakeMotor", 8)
14     optical = Optical("Optical", 11)
15     gps = GPS("GPS", 20)
16
17     #endregion VEXcode Generated Robot Configuration
18     myVariable = 0
19     triballsPickedUp = [ "Triball5", "Triball6", "Triball7", "Triball8", "Triball9" ]
  #this is an array that contains the triballs used for the OODA Loops
20
21
22     """
23     Programmers: Anthony and Shubh
24     Starting Location: E
25     Starting Direction: North
26     Starts with robot preload
27     Field Preload Location: 2
28     """
29
30
31     #this is the definition of where all triballs are using the GPS sensor
32     Triball1Loc=(-1600,1600)#top left
33     Triball2Loc=(-1550,-1550)#bottom left
34     Triball3Loc=(-600,0)#middle left
35     Triball4Loc=(0,1500) #top middle
36     Triball5Loc=(-125,1075) #from this point these are the five triballs slightly lef
  t of middle
37     Triball6Loc=(-125,550)
38     Triball7Loc=(-100,0)
39     Triball8Loc=(-100,-550)
40     Triball9Loc=(-100,-1050)
41     Triball10Loc=(0,-1500)#Bottom middle
42     Triball11Loc=(1400,1450)#top right
43     Triball12Loc=(1450,-1455)#bottom right
44     LoadingZone1=(-1400,1450)
45     LoadingZone2=(-1400,-1450)
46
47     #target points
48     tp1=(1200,300)
49     tp2=(1200,0)
50     tp3=(1200,-300)
51     listOfTargetPoints=[[tp1,1],[tp2,0],[tp3,0]]
52
```

```
53      def when_started1():
54          #This section sets the velocity of all motors to 100% and defines its startin
    g heading as 0
55          arm_motor.set_velocity(100,PERCENT)
56          intake_motor.set_velocity(100,PERCENT)
57          drivetrain.set_drive_velocity(100,PERCENT)
58          drivetrain.set_turn_velocity(100,PERCENT)
59          drivetrain.set_heading(0,DEGREES)
60
61
62          #any of the times that imperial is used is to prevent conversion error when wo
    rking with the tiles(48in) as units
63
64          #this section scores the preloads first in the robot and second at point 2 ()
65          arm_motor.spin_for(FORWARD,1200,DEGREES,wait=False)
66          drivetrain.drive_for(FORWARD,70,INCHES,wait=True)
67          drivetrain.turn_to_heading(270,DEGREES,wait=True)
68          intake_motor.spin_for(REVERSE,360,DEGREES,wait=True)
69          drivetrain.turn_to_heading(0,DEGREES,wait=True)
70          drivetrain.drive_for(FORWARD,12,INCHES,wait=True)
71          intake_motor.spin_for(FORWARD,290,DEGREES,wait=False)
72          drivetrain.drive_for(REVERSE,12,INCHES,wait=True)
73          drivetrain.turn_to_heading(270,DEGREES,wait=True)
74          intake_motor.spin_for(REVERSE,360,DEGREES,wait=True)
75
76
77          #this this grabs the triball in the center front of the blue goal
78          drivetrain.turn_to_heading(145,DEGREES,wait=True)
79          intake_motor.spin(FORWARD)
80          drivetrain.drive_for(FORWARD,80,MM,wait=True)
81          drivetrain.drive_for(REVERSE,80,MM,wait=False)
82          intake_motor.stop()
83          drivetrain.turn_to_heading(90,DEGREES,wait=True)
84          drivetrain.drive_for(FORWARD,(42)/3,INCHES,wait=True)
85          intake_motor.spin_for(REVERSE,420,DEGREES,wait=False)
86          drivetrain.drive_for(FORWARD,((2*42)/3),INCHES,wait=True)
87
88          #this is the OODA Loop it iterates to grab the 5 center triballs
89          i=1
90          while i <= 5:
91              OODA()
92              i+=1
93
94          #this code pickes up the triball under the red horizantal elevation bar
95          vector=distanceToPoint((900,-1500))
96          drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
97          drivetrain.drive_for(FORWARD,vector[0]+341,MM,wait=True)
98          vector=distanceToPoint(Triball10Loc)
99          drivetrain.turn_to_heading(vector[1]-1,DEGREES,wait=True)
100         pickUp(vector[0],MM)
101
102         #this code places the triabll in to the side of the goal
103         vector=distanceToPoint((900,-1500))
104         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
```

```
105         drivetrain.drive_for(FORWARD,vector[0]+341,MM,wait=True)
106         vector=distanceToPoint((1500,-550))
107         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
108         intake_motor.spin_for(REVERSE,360,DEGREES,wait=False)
109         drivetrain.drive_for(FORWARD,vector[0]+100,MM,wait=True)
110         drivetrain.drive_for(REVERSE,341,MM,wait=True)
111 
112         #this code picks up and drops off the triball in the lower blue loading zone
113         vector=distanceToPoint(Triball12Loc)
114         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
115         pickUp(vector[0]+341,MM)
116         vector=distanceToPoint((1650,-550))
117         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
118         drivetrain.drive_for(FORWARD,300,MM,wait=True)
119         intake_motor.spin_for(REVERSE,360,DEGREES,wait=False)
120         drivetrain.drive_for(FORWARD,300,MM,wait=True)
121         wait(50,MSEC)
122         drivetrain.drive_for(REVERSE,350,MM,wait=True)
123 
124         #this section returns to a point where it can go straight to the bottom red lo
  ading zone
125         vector=distanceToPoint((900,-1500))
126         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
127         drivetrain.drive_for(FORWARD,vector[0]+341,MM,wait=True)
128 
129         #pickes up the triball in the lower red loading zone
130         vector=distanceToPoint(((Triball2Loc[0]-150),(Triball2Loc[1]+100)))
131         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
132         drivetrain.drive_for(FORWARD,vector[0],MM,wait=True)
133         intake_motor.spin(FORWARD)
134         drivetrain.turn_for(LEFT,35,DEGREES,wait=True)
135         intake_motor.stop()
136         drivetrain.turn_for(RIGHT,185,DEGREES,wait=True)
137 
138         #launches the triabll from around the center triball position
139         vector=distanceToPoint(Triball7Loc)
140         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
141         intake_motor.spin(FORWARD)
142         arm_motor.spin_for(REVERSE,950,DEGREES,wait=False)
143         drivetrain.drive_for(FORWARD,vector[0]+60,MM,wait=True)
144         intake_motor.stop()
145         drivetrain.turn_to_heading(260,DEGREES,wait=True)
146         drivetrain.drive_for(REVERSE,341,MM,wait=False)
147         intake_motor.spin_for(REVERSE,460,DEGREES,wait=True)
148 
149         #spins the arm down from launchincg position and move to pick up triball in th
  e top red loading zone
150         arm_motor.spin_for(FORWARD,950,DEGREES,wait=False)
151         vector=distanceToPoint(((Triball1Loc[0]-30),(Triball1Loc[1]-50)))
152         drivetrain.turn_to_heading(vector[1]+1,DEGREES,wait=True)
153         pickUp(vector[0]+80,MM)
154         intake_motor.spin(FORWARD)
155 
156         #launches the triabll from around the center triball position
```

```
157         vector=distanceToPoint(Triball7Loc)
158         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
159         intake_motor.spin(FORWARD)
160         arm_motor.spin_for(REVERSE,950,DEGREES,wait=False)
161         drivetrain.drive_for(FORWARD,vector[0],MM,wait=True)
162         intake_motor.stop()
163         drivetrain.turn_to_heading(275,DEGREES,wait=True)
164         drivetrain.drive_for(REVERSE,241,MM,wait=False)
165         intake_motor.spin_for(REVERSE,470,DEGREES,wait=True)
166
167         #this function lines up to the triball under the blue horizantal elevation bar
168         vector=distanceToPoint((-400,1500))
169         arm_motor.spin_for(FORWARD,950,DEGREES,wait=False)
170         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
171         drivetrain.drive_for(FORWARD,vector[0]+300,MM,wait=True)
172
173         #picks up the triball under the blue horizantal elevation bar
174         vector=distanceToPoint(Triball4Loc)
175         intake_motor.spin(FORWARD)
176         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
177         drivetrain.drive_for(FORWARD,vector[0],MM,wait=True)
178
179         #launches triball from about half the distance to the goal in to the goal
180         arm_motor.spin_for(FORWARD,1000,DEGREES,wait=False)
181         vector=distanceToPoint((900,1500))
182         intake_motor.stop()
183         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
184         drivetrain.drive_for(FORWARD,vector[0]+300,MM,wait=True)
185         vector=distanceToPoint((1550,550))
186         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
187         drivetrain.drive_for(FORWARD,vector[0]/2,MM,wait=True)
188         intake_motor.spin_for(REVERSE,480,DEGREES,wait=True)
189
190
191         #grabs the triball in the top blue loading zone and
192         vector=distanceToPoint(Triball11Loc)
193         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
194         pickUp(vector[0]+341,MM)
195
196         #places triball in the goal
197         vector=distanceToPoint((1650,550))
198         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
199         drivetrain.drive_for(FORWARD,320,MM,wait=True)
200         intake_motor.spin_for(REVERSE,360,DEGREES,wait=False)
201         drivetrain.drive_for(FORWARD,300,MM,wait=True)
202         wait(50,MSEC)
203         drivetrain.drive_for(REVERSE,330,MM,wait=True)
204
205         #moves to line up to with the top red loading zone
206         vector=distanceToPoint((900,1500))
207         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
208         drivetrain.drive_for(FORWARD,vector[0]+341,MM,wait=True)
209
210         #picks up triball in the top red loading zone
```

```
211         vector=distanceToPoint(((Triball1Loc[0]-100),(Triball1Loc[1]-100)))
212         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
213         drivetrain.drive_for(FORWARD,vector[0],MM,wait=True)
214         intake_motor.spin(FORWARD)
215         drivetrain.turn_for(RIGHT,35,DEGREES,wait=True)
216         intake_motor.stop()
217         drivetrain.turn_for(LEFT,185,DEGREES,wait=True)
218 
219         #does the final launch around the center triball location
220         vector=distanceToPoint(Triball7Loc)
221         drivetrain.turn_to_heading(vector[1],DEGREES,wait=True)
222         intake_motor.spin(FORWARD)
223         arm_motor.spin_for(REVERSE,950,DEGREES,wait=False)
224         drivetrain.drive_for(FORWARD,vector[0]+60,MM,wait=True)
225         intake_motor.stop()
226         drivetrain.turn_to_heading(285,DEGREES,wait=True)
227         drivetrain.drive_for(REVERSE,341,MM,wait=False)
228         intake_motor.spin_for(REVERSE,470,DEGREES,wait=True)
229         pass
230     vr_thread(when_started1)
231 
232     #this function gets the distance from the current point to the desired position sp
    ecified in the argument
233     def distanceToPoint(DesiredPosition):
234         CurrentPosition=(gps.x_position(MM),gps.y_position(MM))
235         yDistance=DesiredPosition[1]-CurrentPosition[1]
236         xDistance=DesiredPosition[0]-CurrentPosition[0]
237         distanceToPointSloved = math.hypot(yDistance,xDistance)-381
238         angleRad=math.atan2(DesiredPosition[1]-CurrentPosition[1],DesiredPosition[0]-C
    urrentPosition[0])
239         angleDeg=90-math.degrees(angleRad)
240         return distanceToPointSloved, angleDeg
241 
242     #this picks up triballs with a distance argument to drive forward and it takes a i
    nches or millimeters argument so that this can be used with the imperial and metric par
    ts of the program
243     def pickUp(Special,inOmm):
244         if(not(arm_motor.position(DEGREES) == 1200)):
245             arm_motor.spin_for(FORWARD,1200,DEGREES,wait=False)
246         intake_motor.spin(FORWARD)
247         drivetrain.drive_for(FORWARD,Special,inOmm,wait=True)
248         intake_motor.stop()
249 
250     #this is the OODA loop it is the program that grabs and deposits triballs from th
    e center to the goal
251     def OODA():
252         #observe/orient
253         #initializes increment variables and list for storing options
254         workDistance=[]
255         wi = 0
256         i = 0
257 
258         #gets the distance to all triballs currently outside of the goal
259         for triball in triballsPickedUp:
```

```
260             if triball == "Triball9":
261                 workDistance.insert(wi,(triball, *distanceToPoint(Triball9Loc)))
262             elif triball == "Triball8":
263                 workDistance.insert(wi,(triball, *distanceToPoint(Triball8Loc)))
264             elif triball == "Triball7":
265                 workDistance.insert(wi,(triball, *distanceToPoint(Triball7Loc)))
266             elif triball == "Triball6":
267                 workDistance.insert(wi,(triball, *distanceToPoint(Triball6Loc)))
268             elif triball == "Triball5":
269                 workDistance.insert(wi,(triball, *distanceToPoint(Triball5Loc)))
270             wi += 1
271 
272         #decide
273         #figures out the closest point and changes the triball to "done" so it is igno
    red by the above for loop
274         minimum=min(workDistance, key=lambda x: x[1])
275         if triball in triballsPickedUp:
276                 index = triballsPickedUp.index(minimum[0])
277                 triballsPickedUp[index] = "done"
278 
279 
280         #act
281         #turns to the heading and grabs the closest triball
282         drivetrain.turn_to_heading(minimum[2]-2,DEGREES,wait=True)
283         intake_motor.spin(FORWARD)
284         drivetrain.drive_for(FORWARD,minimum[1]+70,MM,wait=True)
285         intake_motor.stop()
286 
287 
288         #does the same thing but for the target
289         #initializes increment variables and list for storing options
290         workingTargetList=[]
291         wit=0
292 
293         #gets the distance to all target points in the goal
294         for i in listOfTargetPoints:
295 
296             workingTargetList.insert(wit,(i, *distanceToPoint(i[0])))
297             wit=1+wi
298 
299         #sorts the working list by the distance lowest to highest
300         minimumTWork=sorted(workingTargetList, key=lambda x: x[1])
301 
302         #uses the closest target point
303         minimumT=minimumTWork[0]
304 
305         #checks whether the chosen target is full (3 or more triballs) if it is full i
    t chooses the second lowest
306         if i in listOfTargetPoints:
307                 index = listOfTargetPoints.index(minimumT[0])
308                 if listOfTargetPoints[index][1] == 3:
309                     minimumT = minimumTWork[1]
310                 else:
311                     listOfTargetPoints[index][1] += 1
```

```
312
313         #turns to the heading and moves towards the goal
314         drivetrain.turn_to_heading(minimumT[2],DEGREES,wait=True)
315         driveDistance=(minimumT[1])
316         #the drive train drives to the goal< after a third of the distance is covered
  it starts reversing the intake then completes the distance
317         drivetrain.drive_for(FORWARD,(driveDistance+25)*(1/3),MM,wait=True)
318         intake_motor.spin_for(REVERSE,470,DEGREES,wait=False)
319         drivetrain.drive_for(FORWARD,(driveDistance+25)*(2/3),MM,wait=True)
320         drivetrain.drive_for(REVERSE,50,MM,wait=True)
321
322
323
324
325
```