



4.6 Project: the marker trail

Vision · Robot club · about 25 min

Name _____

Class _____

Date _____

What this lesson is about

Visit three tag cubes in order.

- Cubes, not cards
- The two functions
- The trail is a loop
- Where next

Questions

6 marks in all

1. Why are the markers in this project on cubes instead of cards? [1 mark]

- ☐ A There is a tag on every side, so they can be read from any direction
- ☐ B Cubes are easier for the blob detector
- ☐ C Cards cannot hold a number above 1
- ☐ D The robot can push a cube out of the way

2. What does this program print? [1 mark]

```
tags = [[2, 100, 120, 40.0], [1, 250, 118, 65.0]]
print([t[0] for t in tags])
```

3. Put the three lines of the marker trail in order. [1 mark]

Number the lines 1 to 3 to put them in the right order.

```
___ `for tag_id in [1, 2, 3]:`
___ ` find(tag_id)`
___ ` approach(tag_id, 10)`
```

4. You want to visit the markers in the order 3, 1, 2. What do you change? [1 mark]

- ☐ A Just the list: [3, 1, 2]
- ☐ B Rewrite `find` for each marker
- ☐ C Rewrite `approach` for each marker
- ☐ D Add a new `set_cv` for each marker

5. What does this program print?

[1 mark]

```
steps = []

def find(tag_id):
    steps.append("find " + str(tag_id))

def approach(tag_id, stop_at):
    steps.append("go " + str(tag_id))

for tag_id in [3, 1, 2]:
    find(tag_id)
    approach(tag_id, 10)
print(", ".join(steps))
```

6. A student gets marker 1 working, then copies the whole search and approach code out twice more for markers 2 and 3. What is the main problem? [1 mark]

- ☐ A A fix to one copy has to be made in all three, and it is easy to miss one
- ☐ B Python cannot run the same code three times
- ☐ C The camera can only find one marker per program
- ☐ D Copies run more slowly than functions

The task: the marker trail

Visit markers 1, 2 and 3 in order, stopping close to each. Sixty seconds.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

def bearing_of(cx):
    return (cx - 160) * 120 / 320

def find(tag_id):
    # again and again, for ever
    while True:
        tags = [t for t in apriltags() if t[0] == tag_id]
        if tags:
            # leave the loop
            break
        # spin clockwise on the spot at 40
        turn_right(40)
        # pause 0.1 s (the robot keeps doing what it was told)
        wait(0.1)
    # all motors off
    stop()
    # pause 0.3 s (the robot keeps doing what it was told)
    wait(0.3)

def approach(tag_id, stop_at):
    # again and again, for ever
    while True:
        tags = [t for t in apriltags() if t[0] == tag_id]
        if not tags:
            # spin clockwise on the spot at 30
            turn_right(30)
            # pause 0.1 s (the robot keeps doing what it was told)
            wait(0.1)
            # back to the top of the loop
            continue
        cx, dist = tags[0][1], tags[0][3]
        if dist <= stop_at:
            # leave the loop
            break
        speed = max(20, min(70, (dist - stop_at) * 3))
        # forward, sideways, rotation: -100 to 100 each, until the next command
        drive(speed, 0, bearing_of(cx) * 3)
        # pause 0.1 s (the robot keeps doing what it was told)
        wait(0.1)
    # all motors off
    stop()

# camera: the tag detector
set_cv("apriltag")
find(1)
approach(1, 10)
```

Plan your program here, then type it in and press Run.



Do it on the robot

www.bugbotlab.com/learn/4-6-project-marker-trail/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Visit them in the order 3, 1, 2. Which order is fastest?
2. Print the time when you reach each marker, using a variable that counts your `wait` calls or the `t` from the telemetry.
3. Make `find` remember which way it turned to find the last tag and search that way first.