WORKSHEET

9.6 Sharing what you see

Talking to each other · Robot club · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

Telling a robot without a camera where the ball is.

- Where is the ball?
- Say what, not what you see

Questions

6 marks in all

1. Which message is useful to the Fetcher, which is standing somewhere else? [1 mark]

- ☐ **A** `ball at 20,30`
- ☐ **B** `red blob at cx 210, 40 px wide`
- ☐ **C** `ball in my camera`
- ☐ **D** `turn right 15`

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

```
import math

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

x, y, heading = 10, 0, 0
cx, width = 160, 23
dist = 4 * 92 / max(width, 1)
h = math.radians(heading + bearing_of(cx))
bx, by = x + math.sin(h) * dist, y + math.cos(h) * dist
print(f"ball at {bx:.0f},{by:.0f}")
```

3. The blob's `cx` is 200. Using `bearing_of(cx) = (cx - 160) * 120 / 320`, what is its bearing in degrees? [1 mark]

4. Why is the direction to the ball `heading() + bearing_of(cx)`? [1 mark]

- ☐ **A** The bearing is measured from where the camera points, so adding the heading turns it into a direction on the mat
- ☐ **B** To make the number positive
- ☐ **C** Because the camera is on the back of the robot
- ☐ **D** The heading is always zero

5. Why does the code use `max(width, 1)` when working out the distance? [1 mark]

- ☐ **A** So it never divides by zero
- ☐ **B** So the ball is never closer than 1 cm
- ☐ **C** To round the width to a whole number
- ☐ **D** Because the width is measured in metres

6. The robot faces up the mat and the red ball is right of centre in the picture. Roughly where is the ball? [1 mark]

- ☐ **A** Up the mat and to the right, at larger x
- ☐ **B** Up the mat and to the left, at smaller x
- ☐ **C** Behind the robot
- ☐ **D** Exactly straight ahead

The task: share what you see

Find the red ball with the camera, work out where it is, and send `ball at <x>,<y>` within 10 cm of the truth. The Fetcher does the rest.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
# camera: look for red blobs
set_cv('blob', 'red')
# until a blob of that colour is in view
while not blobs():
    # 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()
print(blobs())
```

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

Do it on the robot

www.bugbotlab.com/learn/9-6-sharing-what-you-see/

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

Challenges

1. Send the distance from the Fetcher to the ball as well (the Fetcher starts 38 cm to your right and 8 cm behind you).
2. Add a second ball of another colour in free play and report both.
3. Keep watching: if the ball is pushed, send the new position.