WORKSHEET

6.2 Following a line

Seeing more · Robot club · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

cx to rotation, the angle as a look-ahead, speed on the straights.

- One step
- The whole line
- Using the angle
- Speed on the straights

Questions

7 marks in all

1. `follow_step` uses rotation `(cx - 160) * 0.4`. What rotation does a line at `cx = 100` give? [1 mark]

2. The rotation for a line at `cx = 100` is negative. What does the robot do? [1 mark]

- ☐ **A** Turns anticlockwise, towards the line on its left
- ☐ **B** Turns clockwise, away from the line
- ☐ **C** Slides left without turning
- ☐ **D** Stops, because the error is negative

3. A student writes `drive(60, 0, (160 - cx) * 0.4)`. What happens? [1 mark]

- ☐ **A** The robot steers away from the line and loses it
- ☐ **B** It follows the line more smoothly
- ☐ **C** It follows the line backwards
- ☐ **D** Nothing changes, the order does not matter

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

```
def rotation(cx, angle):
    return (cx - 160) * 0.3 + angle * 1.5

print(rotation(160, 10))
print(rotation(200, -8))
```

5. Adding `angle * 1.5` to the rotation is the beginning of which kind of control? [1 mark]

- ☐ **A** Derivative: it reacts to where the error is heading
- ☐ **B** Bang-bang: full on or full off
- ☐ **C** Integral: it adds up past errors
- ☐ **D** Open loop: it ignores the sensor

6. At each bend the line swings well off-centre, up to 85 pixels, before the robot has turned. What does that mean? [1 mark]

- ☐ **A** Nothing is wrong: a P controller only turns once there is an error, and the line is still in view
- ☐ **B** The camera is badly aligned
- ☐ **C** The gain is negative
- ☐ **D** The robot has lost the line

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

```
for angle in [0, 5, 20, -25]:
    speed = 80 if abs(angle) < 10 else 40
    print(angle, speed)
```

The task: follow the line

Follow the line to the green zone, staying within 6 cm of it for at least 85% of the run.

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

# drive forward at 60 for 90 cm, then stop
forward(60, distance=90)
```

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

Do it on the robot

www.bugbotlab.com/learn/6-2-following-a-line/

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

Challenges

1. Make the speed depend on the angle: 80 when the line is straight, 40 in a bend.
2. Print the largest `cx` error you saw during the run.
3. Follow the line backwards from the end to the start (turn round first).