WORKSHEET

6.3 Losing the line

BugBotLab

Seeing more · Robot club · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

Gaps: carry on, then search with a state machine.

- Keep going first
- Then search
- Choosing the first direction

Questions

7 marks in all

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

```
readings = [[160, 0], [], [], [170, 2], [], [], [], []]
lost = 0
for seen in readings:
    if seen:
        lost = 0
    else:
        lost += 1
print(lost)
```

2. When the line disappears, why carry straight on for a while before searching? [1 mark]

- ☐ **A** Most gaps are short, so the line is usually just ahead
- ☐ **B** The camera needs time to switch detector
- ☐ **C** Searching uses more battery
- ☐ **D** The line always restarts straight ahead

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

```
for sweep in [0, 14, 15, 29, 30, 45]:
    print(sweep, 40 if (sweep // 15) % 2 == 0 else -40)
```

4. The search slides from side to side instead of turning. Why? [1 mark]

- ☐ **A** The robot keeps its heading, so it already points the right way when it finds the line
- ☐ **B** Sliding is faster than turning
- ☐ **C** The line detector cannot work while turning
- ☐ **D** Turning would make `line()` return an error

5. Which way should the search try first? [1 mark]

- ☐ **A** The way the last `angle` said the line was heading
- ☐ **B** Always right
- ☐ **C** Always left
- ☐ **D** Backwards, to where the line was

6. The follower switches to search when `lost > 8`, adding 1 to `lost` on each tick with no line. On which tick without a line does it switch? [1 mark]

---

7. The task starter does `if not follow_step(60, 0.4): break`. What happens at the gap? [1 mark]

- ☐ **A** The robot stops at the first gap and never reaches the zone
- ☐ **B** It searches for the line
- ☐ **C** It follows the line through the gap
- ☐ **D** It turns round

The task: find the line again

Follow the line through the gap to the green zone, staying within 6 cm of it for at least 60% of the run.

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

def follow_step(speed=60, gain=0.4):
    # [cx, angle] of the line ahead, or [] if none
    seen = line()
    if not seen:
        return False
    cx, angle = seen
    # forward, sideways, rotation: -100 to 100 each, until the next command
    drive(speed, 0, (cx - 160) * gain)
    return True

# camera: the line detector
set_cv("line")
while position()[1] < 85:
    if not follow_step(60, 0.4):
        # leave the loop
        break
    # pause 0.1 s (the robot keeps doing what it was told)
    wait(0.1)
# all motors off
stop()
```

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

Do it on the robot

www.bugbotlab.com/learn/6-3-losing-the-line/

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

Challenges

1. Sweep in the direction of the last angle you saw.
2. Count how many ticks the search took and print it.
3. Make the sweep widen each time: 10 ticks, then 20, then 30.