ANSWERS

6.6 Bumps and stalls

Seeing more · Robot club · about 20 min

What this lesson is about

bumped(), stalls from velocity, back off and go round.

- bumped()
- Stalls
- Back off, go round
- Wall or robot?

Questions

7 marks in all

1. The robot drives into a 2 cm tall box, but `distance()` never saw it. Why? [1 mark]

- ☐ **A** The depth sensor's level rows look straight over the top of it
- ☐ **B** The box is the same colour as the mat
- ☐ **C** `distance()` only works when the robot is still
- ☐ **D** The camera was switched to the line detector

> **Answer:** A. Low things are invisible to a sensor looking level. The first the robot knows is the bump.

2. Which sensor does `bumped()` use? [1 mark]

- ☐ **A** The accelerometer in the IMU, which feels the jolt
- ☐ **B** The camera
- ☐ **C** The distance sensor
- ☐ **D** The optical-flow sensor

> **Answer:** A. A collision is a sudden jolt, and the accelerometer feels it. `bumped()` is true for a moment afterwards.

3. How does the stall check know the robot is stuck? [1 mark]

- ☐ **A** It compares the real speed from `velocity()` with the speed it asked for
- ☐ **B** It waits for `bumped()`
- ☐ **C** It checks whether `distance()` is under 5
- ☐ **D** It counts how long the program has run

> **Answer:** A. Told to drive at 60 and moving at almost nothing means something is in the way.

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

```
speeds = [0, 10, 25, 30, 29, 28, 1, 0]
for tick, vy in enumerate(speeds):
    if tick > 5 and vy < 2:
        print("stalled at tick", tick)
        break
```

Answer:

```
stalled at tick 6
```

Tick 0 is slow too, but `tick > 5` ignores the start while the robot gets up to speed. The first real stall is tick 6.

5. Which of these can a stall check catch that `bumped()` might miss? [1 mark]

Tick every answer that is true.

- ☐ **A** A slow push into a wall
- ☐ **B** Driving off the edge of a table onto nothing
- ☐ **C** A tag 30 cm ahead
- ☐ **D** Another robot's LED changing colour

Answer: A, B. A slow collision gives no jolt, and a robot hanging off an edge feels nothing. In both, the flow sensor sees no motion.

6. Put the states of the bump and back behaviour in the order they run after a bump. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `go: drive forward until `bumped()``

___ `back: drive backward for a few ticks`

___ `side: slide right for a few more ticks`

___ `go: carry on forward`

Answer:

```
go: drive forward until `bumped()`
back: drive backward for a few ticks
side: slide right for a few more ticks
go: carry on forward
```

A bump starts a small behaviour that runs to completion, like the avoid layer in lesson 5.4.

7. After a bump, a tag is right in front of you. What is the polite response? [1 mark]

- ☐ **A** It is probably a robot, so wait rather than shove
- ☐ **B** It is a wall, so back off and go round
- ☐ **C** Drive harder to push through
- ☐ **D** Turn the camera off

Answer: A. A wall stays put, a robot moves. Give it a second and the way may clear.

The task: bump and back

Reach the green zone past a box the depth sensor cannot see, printing `bumped` when you hit it.

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

while position()[1] < 80:
    # drive forward at 60 (keeps going until the next command)
    forward(60)
    # pause 0.1 s (the robot keeps doing what it was told)
    wait(0.1)
# all motors off
stop()
```

The hint students can ask for: There is a box the depth sensor cannot see (it is too low). Drive for the goal; when bumped() says you hit it, print `bumped`, back off, go round, and carry on.

A solution

```
from bugbot import *
connect()
state = 'go'
ticks = 0
while position()[1] < 80:
    if state == 'go':
        forward(60)
        if bumped():
            print('bumped')
            state = 'back'
            ticks = 0
    elif state == 'back':
        backward(50)
        ticks += 1
        if ticks > 8:
            state = 'side'
            ticks = 0
    elif state == 'side':
        right(60)
        ticks += 1
        if ticks > 18:
            state = 'go'
    wait(0.1)
stop()
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.