WORKSHEET

5.2 States

Behaviours · Robot club · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

A plan as a state variable, transitions, drawing the machine.

- A state is a variable
- Sliding to a point
- The machine
- Drawing it

Questions

7 marks in all

1. In a state machine, what is the state? [1 mark]

- ☐ **A** A variable holding a word that says which part of the plan the robot is in
- ☐ **B** The robot's position on the mat
- ☐ **C** The last sensor reading
- ☐ **D** A function that drives the robot

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

```
import math

route = {"up": (0, 60), "right": (60, 60)}
state = "up"
for p in [(0, 20), (0, 57), (30, 60), (57, 61)]:
    tx, ty = route[state]
    if math.hypot(tx - p[0], ty - p[1]) < 6:
        if state == "right":
            print("done")
            break
        state = "right"
    print(p, state)
```

3. In the `"up"` branch of a patrol, the transition is `if near(0, 60): state = "up"`. The robot reaches (0, 60) and just sits there. Why? [1 mark]

- ☐ **A** The transition sets the state back to `"up"`, so it never leaves that state
- ☐ **B** `near` needs a `cm` argument
- ☐ **C** `go_to` cannot drive to (0, 60)
- ☐ **D** The loop is missing `wait(0.1)`

4. `target = route[state]` gives a pair like `(60, 60)`. What does `near(*target)` do? [1 mark]

- ☐ **A** Unpacks the pair into two arguments, the same as `near(60, 60)`
- ☐ **B** Multiplies the target by something
- ☐ **C** Passes the pair as one argument
- ☐ **D** Gives the distance to the target

5. The task's patrol goes round the box and back. Put its states in order. [1 mark]

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

___ `up`
___ `right`
___ `down`
___ `left`

6. You draw a state machine before writing it. Which belong in the drawing? [1 mark]

Tick every answer that is true.

- ☐ **A** A circle for each state
- ☐ **B** An arrow for each transition
- ☐ **C** The condition written on each arrow
- ☐ **D** A box for every tick of the loop

7. Why does the patrol keep the corners in a dictionary, `route`, instead of writing a separate driving block for each state? [1 mark]

- ☐ **A** The loop body is the same for every state, and only the transitions differ
- ☐ **B** Dictionaries make the robot faster
- ☐ **C** `go_to` only accepts a dictionary
- ☐ **D** A state machine is not allowed to use `if`

The task: patrol in states

From A to B and back, round the box: up, right, then left, down. Without touching the box.

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

# maths: atan2, hypot, sin, cos, radians
import math

def wrapped(h):
    return (h + 180) % 360 - 180

def go_to(x, y, speed=70):
    # where am I?
    px, py = position()
    a = math.radians(wrapped(math.degrees(math.atan2(x - px, y - py)) - heading()))
    # forward, sideways, rotation: -100 to 100 each, until the next command
    drive(speed * math.cos(a), speed * math.sin(a), wrapped(0 - heading()) * 3)

def near(x, y, cm=6):
    # where am I?
    px, py = position()
    return math.hypot(x - px, y - py) < cm

# do this 100 times (tick counts from 0)
for tick in range(100):
    if near(0, 60):
        # leave the loop
        break
    go_to(0, 60)
    # 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/5-2-states/

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

Challenges

1. Add a "pause" state: stop for one second at B before coming back.

2. Patrol twice, with a lap counter.
3. Make the LED a different colour in each state.